Final Review

Dr. Rafal A. Angryk

Outline

- Covered readings in textbooks
- Regular type of test on the final (single choice!)
- A few exercises
  - ER
  - Normalization

Readings in textbook (1)

- Part 1 Introduction and Conceptual Modeling
  - Chapter 1.1-6: Databases and database Users
  - Chapter 2.1-7: System concepts and Architecture
  - Chapter 3.1-9: Modeling using the E-R diagram

Readings in textbook (2)

- Part 2 The Relational Model
  - Chapter 5: Concepts and Constraints
  - Chapter 6.1-7: Relational Algebra
  - Section 7.1-3: Mapping ER diagrams to tables
  - Chapter 8.1-8, 9.: SQL schema definition, constraints, queries, views

Readings in textbook (3)

- Part 3 Database Design
  - Chapter 10.1-6: Functional dependencies and normalization
  - Chapter 11.Intro: Two approaches to normalization

Readings in textbook (4)

- Part 4 Data storage, indexing, query processing
  - Chapter 13: Disk storage, file structures, & hashing
  - Chapter 14: Indexing structures for files
  - Section 15.1-3, 15.7: Query optimization & algorithms for join
  - Chapter 16 Practical design & tuning

Readings in textbook (5)

- Part 5 Transaction Processing concepts
  - Chapter 19-19.4: Database recovery techniques

This time - regular type of test for the theory part

- Good news:
  - Just a single answer will be correct
  - Strategy: CHOOSE THE BEST ANSWER
- Bad news:
  - Still tricky… Otherwise it would be trivial!
- Why? There is no way you can be good in practice, if you do not have strong foundations in theory
- As usually there will be more practice than theory

Be sure to practice at home:

- Prepare ER Design in Min, Max notation, and map it to the Relational Diagram
- Transform a Universal Relation to 3NF or BCNF tables, and find a minimal set of full functional dependencies
- Relational Algebra and SQL Commands
- Query optimization
- Insert and delete data to/from B+-trees
- Recommend the best type of indexing for given attributes and storage formats

Exe. - Model the following System Requirements:

1. The bank is organized into branches. Each branch is located in a particular city and is identified by a unique name. The bank monitors the assets of each branch.
2. Bank customers are identified by their *customer-id* values. The bank stores each customer's name, and the street and city where the customer lives. Customers may have multiple saving accounts and can take out loans. A customer may be associated with a particular banker, who may act as either a loan officer or a personal banker for that customer.

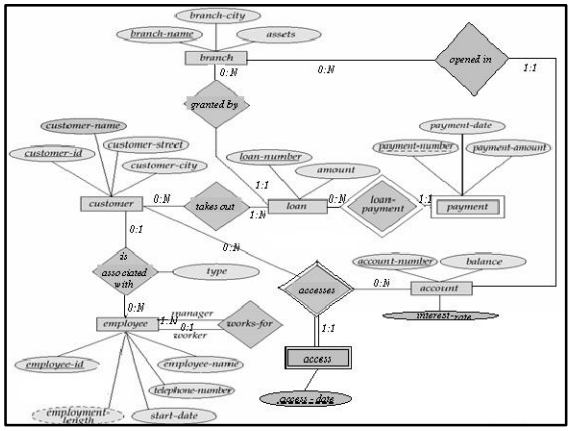
Exe. – cont.

3. Bank employees are identified by their *employee-id* values. The bank administration stores the name and telephone number of each employee, and the *employee-id* number of the employee's manager. The bank also keeps track of the employee's start date and, thus, length of employment.
4. The bank offers only two types of services - saving accounts and loans. Accounts can be held by more than one customer, and a customer can have more than one account. Each account is assigned a unique account number. The bank maintains a record of each account's balance. It also records all dates when the account was accessed by each customer holding the account. Each savings account has a separate interest rate and information about the branch name where it was opened.

Exe. – cont.

5. A loan originates at a particular branch and can be held by one or more customers. A loan is identified by a unique loan number. For each loan, the bank keeps track of the loan amount of the loan payments. Although a loan payment number does not uniquely identify a particular payment among those for all the bank's loans, a payment number does identify a particular payment for a specific loan. The date and amount are recorded for each payment

- Note: Do not worry about making this description complete. Just design data model so it would reflect all requirements mentioned above



Now map it to Relational Model and practice SQL on it

E.g. Retrieve all bank account numbers (only unique values), which were opened in Bozeman, and accessed by Mr. John Smith (i.e. 'John Smith') on 10th of February (i.e. 'Feb/05/2005').

- **Practice optimization of your queries**
- **Put the same in Relational Algebra**
- **What type of indexes would you specify?**
- **Are your tables normalized?**

ER Design Decisions

- **The use of an attribute or entity type (set) to represent an object.**
- **Whether a real-world concept is best expressed by an entity set or a relationship type (set).**
- **The use of a ternary relationship versus a pair of binary relationships.**
- **The use of a strong or weak entity set.**
  - **The use of specialization/generalization – contributes to modularity in the design.**
  - **The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.**

ER Design Crucial Elements

- Use of entity types vs. attributes
  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- Use of entity types (sets) vs. relationship types (sets)
  Possible guideline is to designate a relationship set to describe an action that occurs between entities
- Binary versus *n*-ary relationship types
  Although it is possible to replace any nonbinary (*n*-ary, for *n* > 2) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.
- Appropriate placement of relationship attributes

More on normalization

- **Boyce-Codd Normal Form (BCNF)**
- **One more example of normalization**

Why BCNF is important

- **When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.**
- **3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys**
  - **i.e. composite candidate keys with at least one attribute in common.**

Reminder

- Superkey of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form (3NF)** if whenever a FD X -> A holds in R, then either:
  - (a) X is a superkey of R, or
  - (b) A is a prime attribute of R

NOTE:

Boyce-Codd normal form disallows condition (b) above

What is a Prime Attribute

A prime attribute **must be a** ***member of some candidate key***

A nonprime (or nonkey) attribute is not a prime attribute - that is, it is ***not a member of any candidate key*****.**

In other words …

- A relation is in **3NF** if every nonkey attribute is in 2NF and **nontransitively dependent on a candidate key**
- **BCNF** is based on the concept of a ***determinant*****.**
  - A determinant is any attribute (simple or composite) on which some other attribute is <u>fully</u> functionally dependent
- A relation is in **BCNF** <u>if and only if</u> **every determinant is a candidate key**

  i.e. whenever an FD X -> A holds in R, then X must be a superkey of R

Remember

- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- BCNF is a simpler, yet stricter form of 3NF
- NOTE: There exist relations that are in 3NF but not in BCNF

The theory

- Consider the following relation and FDs: R(a,b,c,d)
  - (1) a,c -> b,d
  - (2) a,d -> b
- *Here, the first determinant suggests that the primary key of R could be changed from a,b to a,c.*
- *If this change was done all of the non-key attributes present in R could still be determined, and therefore this change is legal.*
- *However, the second determinant indicates that a,d determines b, but a,d could not be the key of R as a,d does not determine <u>all</u> of the non key attributes of R (it does not determine c).*
- *We would say that the determinant in the first FD is a candidate key, but the second determinant is not a candidate key, and thus <u>this relation is not in BCNF</u>*
- *However, R is in 3NF since in the second FD b is a prime attribute of R*

Normalization to BCNF – Example (1)

- The CLINIC relation above depicts a special weight-loss surgery clinic where the each patient has 4 appointments:
- On the first they are weighed,
- On the second they are exercised,
- On the third they have gastric surgery,
- And on the fourth they are re-examined and given diet tips to not gain the weight again
- Not all patients need all four appointments!
- Appointment 1 is either 09:00 or 13:00, appointment 2 10:00 or 14:00, and so on…

Normalization to BCNF – Example (2) Universal Table

CLINIC

Patient No	PaienttName	AppointmentNo	Time	Doctor
1	John	1	09:00	Smith
2	Kerr	1	09:00	McCoy
3	Adam	2	10:00	Smith
4	Robert	1	13:00	McCoy
5	Zane	2	14:00	Smith

Normalization to BCNF – Example (3)

- CLINIC (PatNo, PatName, AppNo, Time, Doctor)
- From this scenario DB designer extracted the following determinants:
  a. PatNo -> PatName
  b. PatNo, AppNo -> Time,Doctor
  c. Time -> AppNo
- Two options for 1NF primary key selection:
  1. CLINIC (PatNo, PatName, AppNo, Time, Doctor)
  2. CLINIC (PatNo, PatName, AppNo, Time, Doctor)

  Transitive FD here: (Time-> AppNo), PatNo -> Doctor

First Choice of Primary Key

CLINIC (PatNo, PatName, AppNo, Time, Doctor)

- 1NF - no non-atomic values, so in 1NF
- 2NF – eliminate Redundant Information (i.e. Partial key dependencies):
  - VISIT (PatNo,AppNo,Time,Doctor)
  - PATIENT (PatNo,PatName)
- 3NF – no transient dependences so in 3NF
- Now we test for BCNF

Is EVERY determinant a Candidate Key?

- Go through all determinates where ALL of the left hand side attributes are present in a relation and at least ONE of the right hand side attributes is also present in the relation.

Check for VISIT

- VISIT (PatNo, AppNo,Time,Doctor)
  a) Patno -> PatName
     Patno is present in DB, but not PatName, so irrelevant.
  b) Patno, AppNo -> Time, Doctor
     All LHS and RHS present so relevant. Is this a candidate key? PatNo, AppNo IS the key, so this is a candidate key. OK for BCNF compliance
  c) Time -> AppNo
     Time is present, and so is AppNo, so relevant. Is this a candidate key?

     If it was then we could rewrite VISIT as:
     VISIT (PatNo,AppNo,Time,Doctor)
     This will not work, as you need both Time and PatNo together to form a unique key. Thus this determinate is not a candidate key, and therefore VISIT is not in BCNF.

Now do the same for PATIENT relation

Can we fix the VISIT relation?

- Note: Time is enough to work out the appointment number of a particular patient

  VISIT (PatNo, Time, Doctor)
  PATIENT (PatNo,PatName)
  APPOINTMENT (Time, AppNo)
- Now BCNF is satisfied, and the final relations shown are in BCNF

Second Choice of Primary Key

CLINIC (PatNo, PatName, AppNo, Time, Doctor)

- 1NF - no non-atomic values, so in 1NF
- 2NF – eliminate Redundant Information (i.e. Partial key dependencies):
  - VISIT (PatNo,Time, Doctor)
  - PATIENT (PatNo,PatName)
  - APPOINTMENT (Time, AppNo)
- 3NF – no transient dependences so the same as 2NF
- Now we test for BCNF – we already know that this time 3NF is in BCNF

LOAN		
loan-number	*amount*	*originating-branch*

CUSTOMER				
customer-id	*customer-name*	*customer-street*	*customer-city*	*banker-name*

PAYMENT			
payment-number	*loan-number*	*payment-date*	*payment-amount*

BRANCH		
branch-name	*branch-city*	*assets*

BORROWER	
customer-id	*loan-number*

EMPLOYEE					
employee-id	*employee-name*	*telephone*	*start-date*	*employment-length*	*manager*

DEPOSITS		
customer-name	*account-number*	*access-date*

SAVING ACCOUNT			
account-number	*customer-id*	*interest-rate*	*balance*